Amazon EKS

IRSA(IAM Role for Service Account)

고병수, 이수정

AWS Cloud Support Engineer

Agenda

- SDK의 Credential Chain
- IRSA(IAM Role for Service Account)
- Admission Webhooks
- Deep Dive for IRSA
- IRSA 실습

1. SDK의 Credential Chain

- 기본 자격증명 공급자 체인(Default Credential Provider Chain)

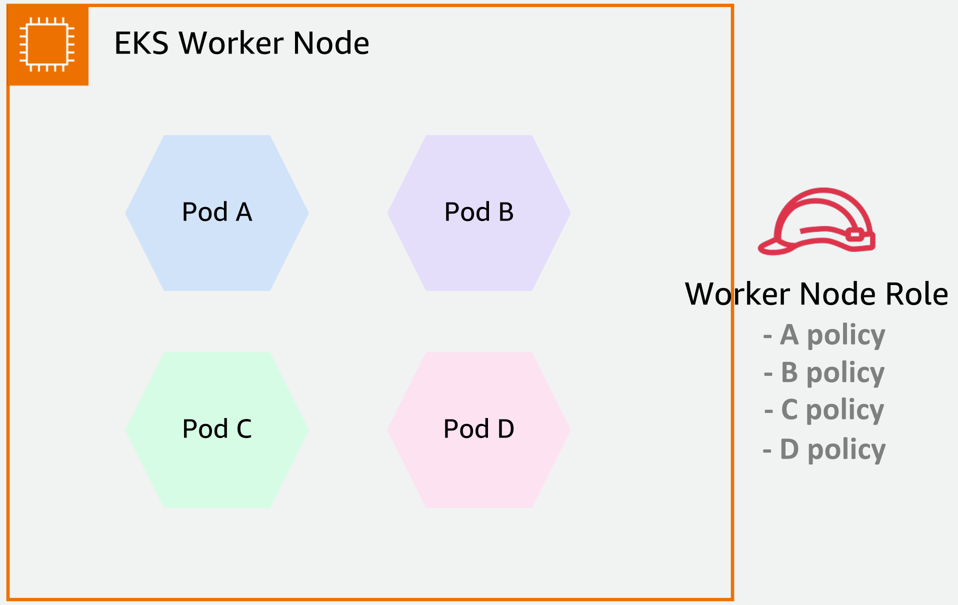
- 자격증명 공급자(Credential Provider)
  - AWS access keys
  - Federate with web identity or OpenID Connect - Assume role credential provider
  - IAM Identity Center credential provider
  - Assume role credential provider
  - Container credential provider
  - Process credential provider
  - IMDS credential provider – EC2 IAM

1. SDK의 Credential Chain

- Java 2.x Credential Provider Chain 순서
  1. Java system properties
  2. Environment Variable
  3. Web identity token from AWS Security Token Service
  4. The shared credentials and config files
  5. Amazon ECS container credentials
  6. Amazon EC2 instance IAM role-provided credentials (Default)

1. SDK의 Credential Chain

- POD에서 실행되는 Application들이 EC2 인스턴스 IAM Role을 사용하게 될 경우
  - 하나의 EC2 인스턴스에(워커노드) A, B, C, D 여러 개의 서로 다른 POD가 동작하고 POD에서 실행되는 Application이 각각 모두 다른 권한을 필요로 할 때 EC2 인스턴스의 IAM Role에는 A Policy, B Policy, C Policy, D Policy 총 4개의 권한이 필요로 하게 되고 다른 Application에서 불필요하게 많은 권한을 가지고 동작할 수 있음



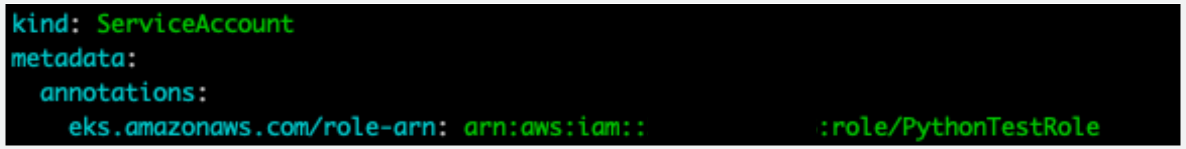
2. IRSA(IAM Role for Service Account)

- Kubernetes Service Account에 IAM Role을 연결하고, POD가 특정 Service Account를 이용하도록 하는 방식

- 장점
  1. 최소권한(Least Privilege)
  2. 자격증명의 격리(Credential Isolation)
  3. 감사(Auditability)

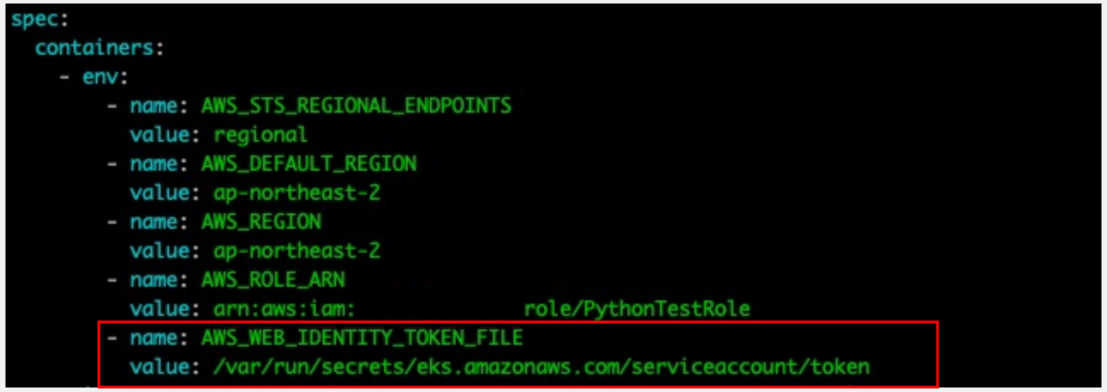
2. IRSA(IAM Role for Service Account)

- IRSA를 이용하는 방법
  1. EKS Cluster에 대한 IAM OIDC Provider 등록
  2. Service Account에 IAM Role 설정



```
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::[redacted]:role/PythonTestRole
```

  3. Pod가 Service Account를 사용하도록 설정
  4. 결과 (POD 환경변수에 Injection)

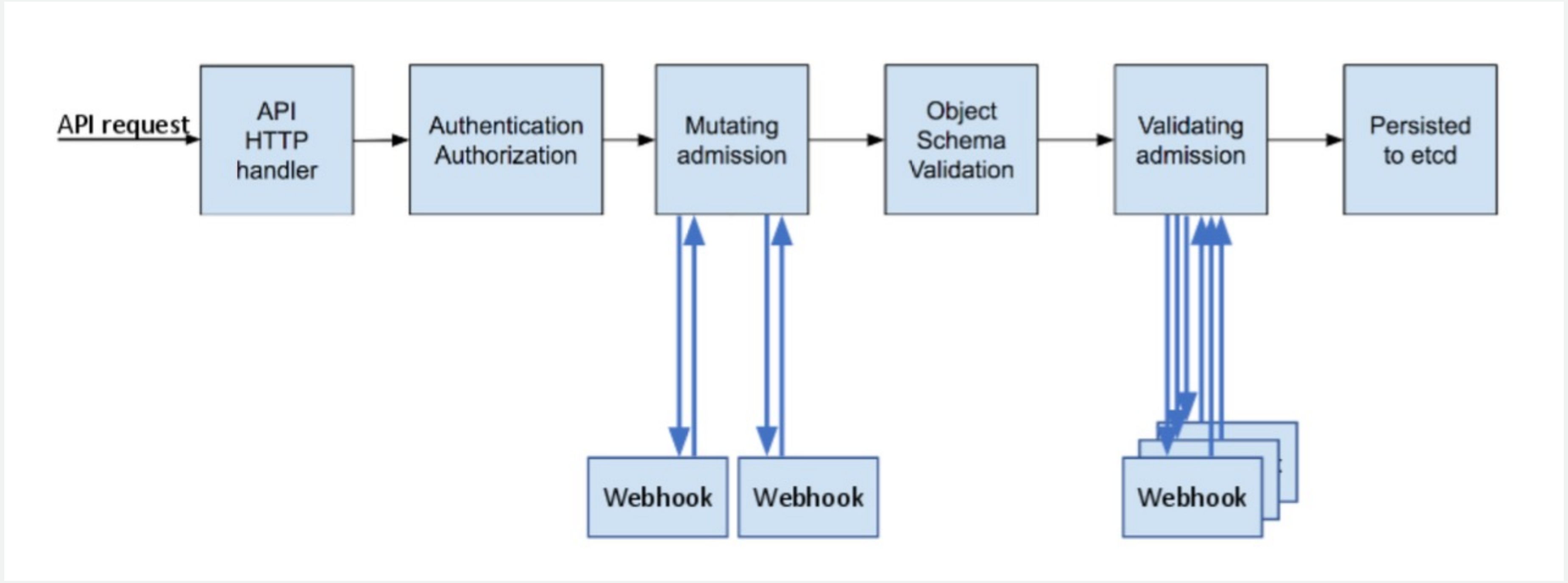


```
spec:
  containers:
    - env:
        - name: AWS_STS_REGIONAL_ENDPOINTS
          value: regional
        - name: AWS_DEFAULT_REGION
          value: ap-northeast-2
        - name: AWS_REGION
          value: ap-northeast-2
        - name: AWS_ROLE_ARN
          value: arn:aws:iam:[redacted]role/PythonTestRole
        - name: AWS_WEB_IDENTITY_TOKEN_FILE
          value: /var/run/secrets/eks.amazonaws.com/serviceaccount/token
```

- Java 2.x Credential Provider Chain
  1. Java system properties
  2. Environment Variable
  3. ★ Web identity token from AWS Security Token Service
  4. The shared credentials and config files
  5. Amazon ECS container credentials
  6. Amazon EC2 instance IAM role-provided credentials (Default)

3. Admission Webhooks

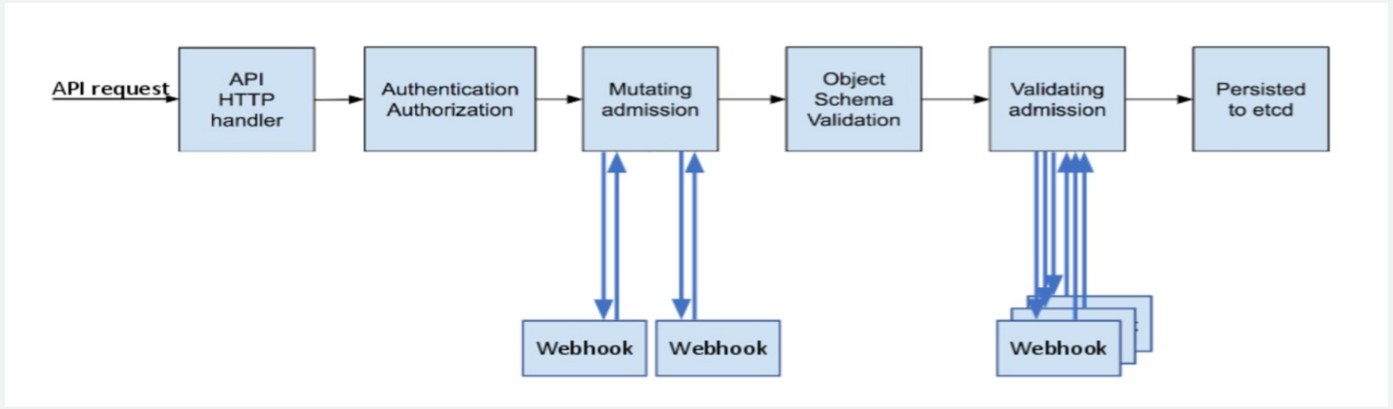
- Admission Webhook은 Admission 요청에 대해 수신하고 이에 대한 작업을 해주는 HTTP Call Back
- Validating Webhook, Mutating Webhook 두 유형이 존재



Admission Controller Phases : https://kubernetes.io/blog/2019/03/21/a-guide-to-kubernetes-admission-controllers/

3. Admission Webhooks

- EKS - Pod Identity Webhook (MutatingWebhookConfiguration 리소스)
  1. CREATE / POD 조건에 의해
  2. url: https://127.0.0.1:23443/mutate 요청
  3. 요청된 URL의 서비스에 의해서 Service Account에 특정 Annotation이 존재하는지 확인
  4. POD에 환경변수 Injection



Admission Controller Phases : https://kubernetes.io/blog/2019/03/21/a-guide-to-kubernetes-admission-controllers/

4. Deep Dive for IRSA

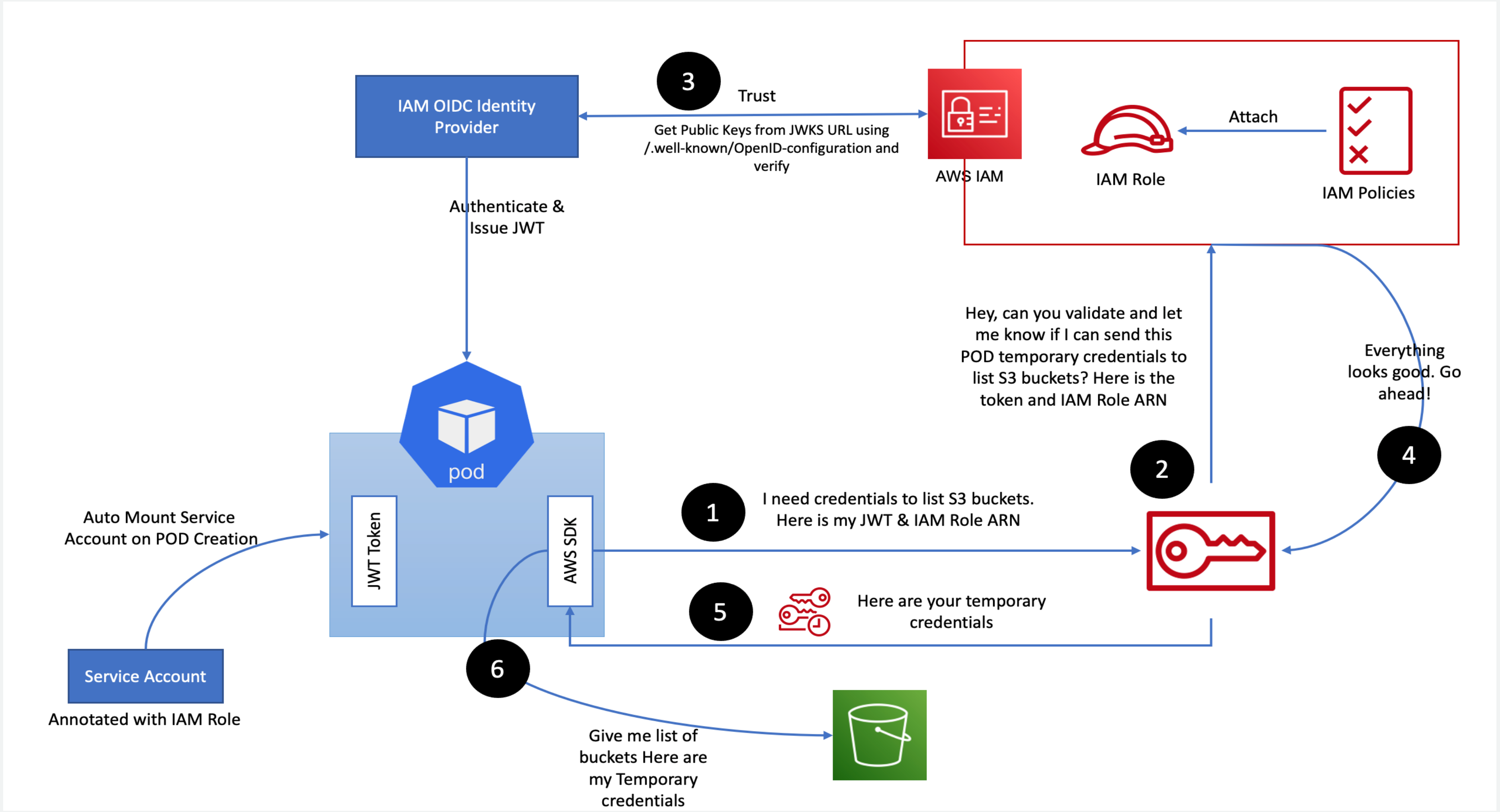
1. IAM서비스에 EKS 클러스터의 OIDC Provider를 Identity Provider로 등록

2. Service Account에 IAM Role을 설정하고, POD는 Service Account를 사용하도록 설정

3. POD 배포

4. pod-identity-webhook(MutatingWebhookConfiguration)에 의해서 POD 환경변수에
   AWS_WEB_IDENTITY_TOKEN_FILE, AWS_ROLE_ARN, AWS_STS_REGIONAL_ENDPOINTS 등이 주입

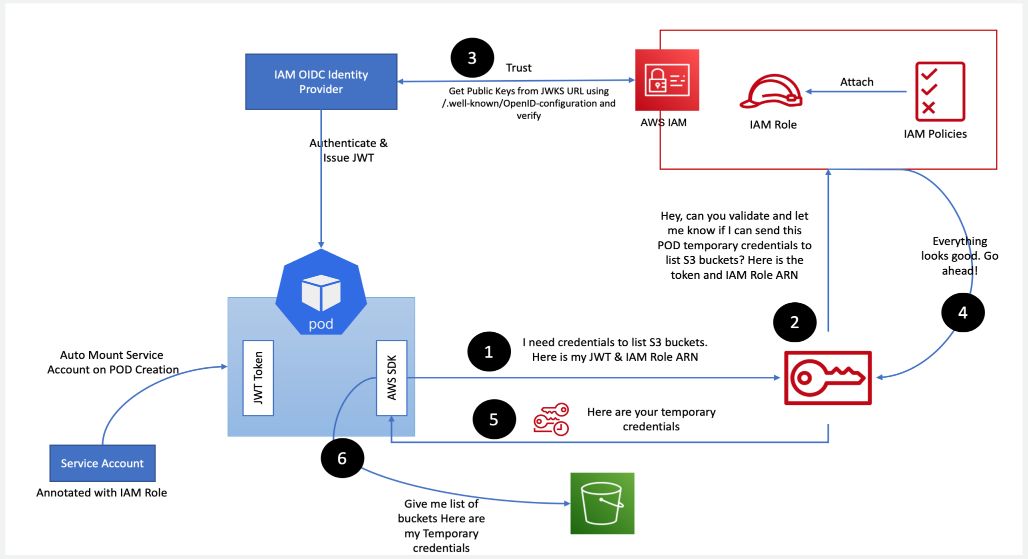
4. Deep Dive for IRSA


IAM OIDC Identity Provider
3
Trust
Get Public Keys from JWKS URL using /.well-known/OpenID-configuration and verify
AWS IAM
IAM Role
Attach
IAM Policies
Authenticate & Issue JWT
Hey, can you validate and let me know if I can send this POD temporary credentials to list S3 buckets? Here is the token and IAM Role ARN
Everything looks good. Go ahead!
4
2
pod
JWT Token
AWS SDK
1
I need credentials to list S3 buckets. Here is my JWT & IAM Role ARN
Auto Mount Service Account on POD Creation
5
Here are your temporary credentials
Service Account
Annotated with IAM Role
6
Give me list of buckets Here are my Temporary credentials

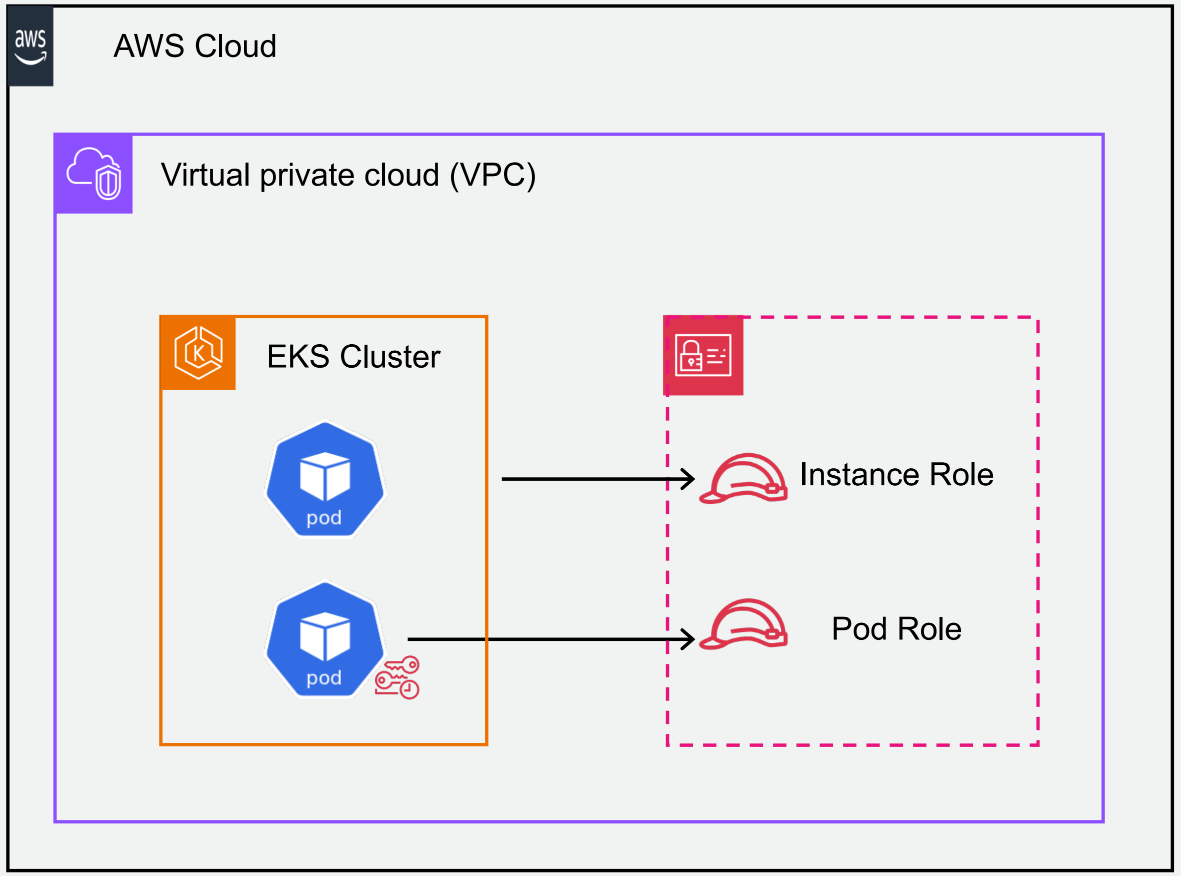
https://aws.amazon.com/ko/blogs/containers/diving-into-iam-roles-for-service-accounts/

4. Deep Dive for IRSA

5. IAM ODIC Identity Provider는 POD에 JWT 발행
6. POD에서 동작하는 Application(SDK)은 JWT 와 IAM Role ARN을 STS로 전달
   - AssumeRoleWithWebIdentity API 호출
7. IAM 서비스는 전달된 JWT가 유효한 토큰인지 한 번 더 검증 후 이상이 없는 경우 임시자격증명이 발급
8. AWS SDK는 발급된 임시 자격증명을 통해 AWS API 호출

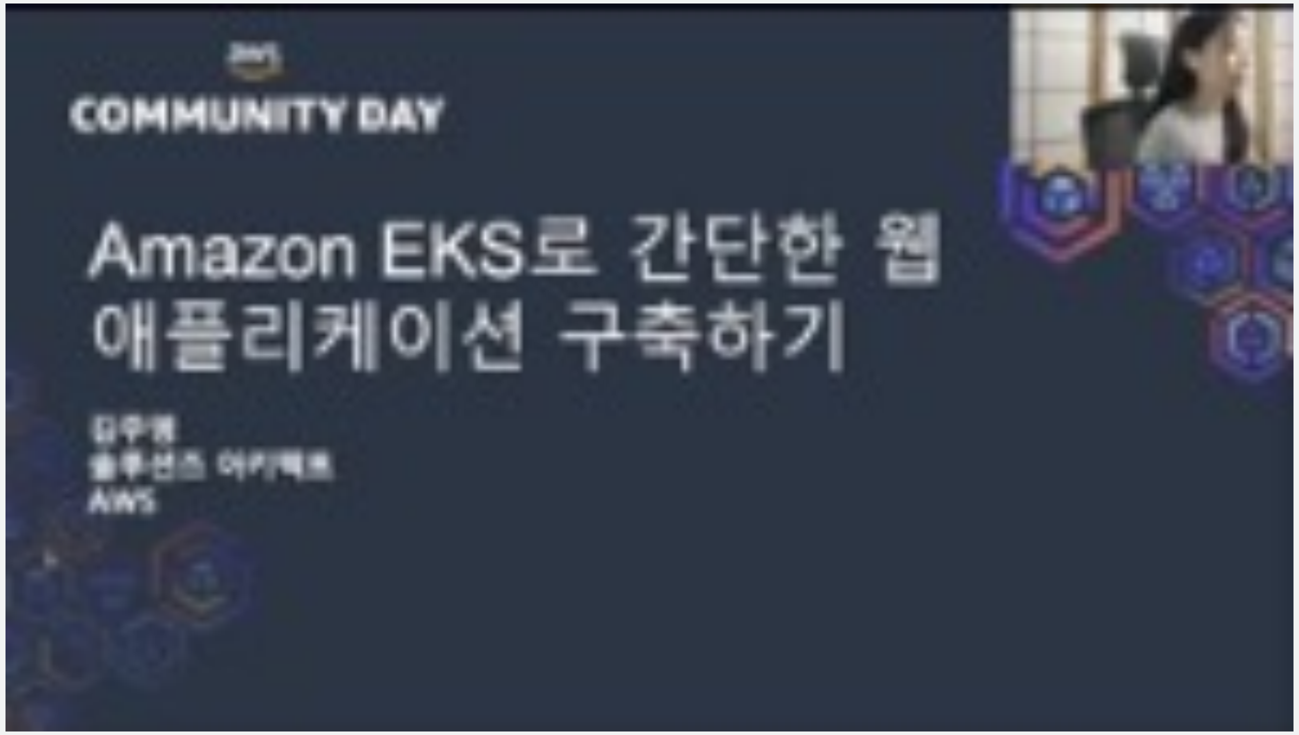

IAM OIDC Identity Provider
3
Trust
Get Public Keys from JWKS URL using /.well-known/OpenID-configuration and verify
AWS IAM
IAM Role
Attach
IAM Policies
Authenticate & Issue JWT
Hey, can you validate and let me know if I can send this POD temporary credentials to list S3 buckets? Here is the token and IAM Role ARN
Everything looks good. Go ahead!
pod
JWT Token
AWS SDK
Auto Mount Service Account on POD Creation
1
I need credentials to list S3 buckets. Here is my JWT & IAM Role ARN
2
4
5
Here are your temporary credentials
6
Service Account
Annotated with IAM Role
Give me list of buckets Here are my Temporary credentials

5. IRSA 실습


AWS Cloud
Virtual private cloud (VPC)
EKS Cluster
pod
pod
Instance Role
Pod Role

5. IRSA 실습

사전 과정 : [Amazon EKS로 간단한 웹 애플리케이션 구성하기]



COMMUNITY DAY

Amazon EKS로 간단한 웹 애플리케이션 구축하기

[illegible]
솔루션즈 아키텍트
AWS

5. IRSA 실습

1. 실습 환경 접속 : [진행시 URL 공유 예정]
2. 실습 가이드 페이지 접속 : [진행시 URL 공유 예정]

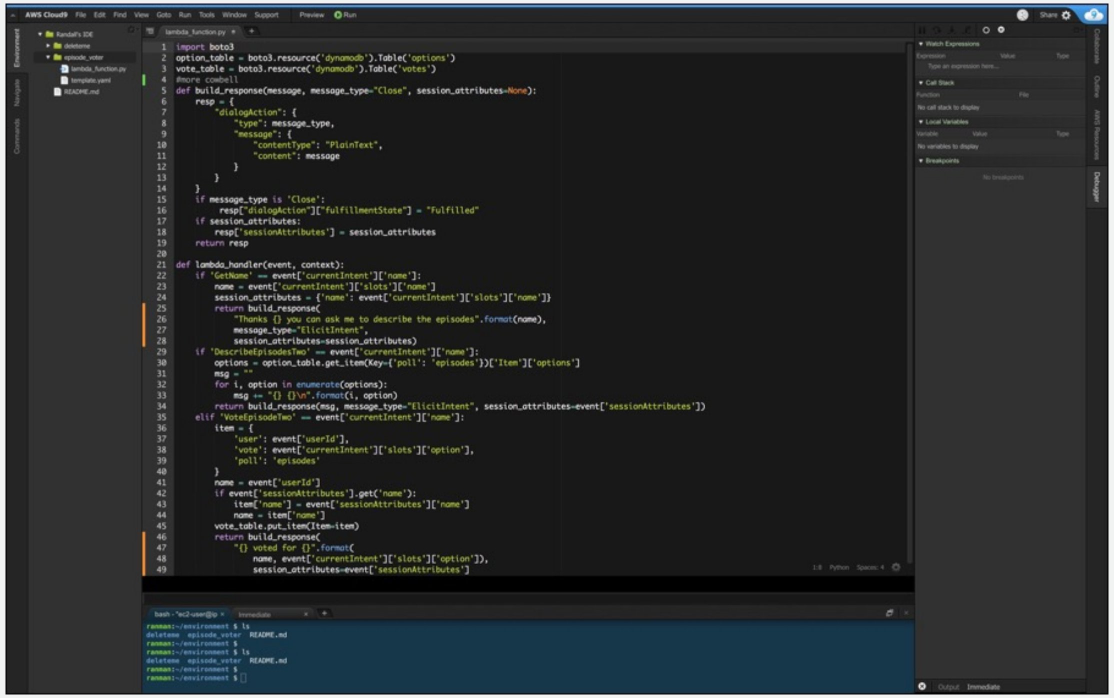
5. IRSA 실습 – Cloud9

What is AWS Cloud9?

✓ A cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser.

✓ Comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, and PHP.

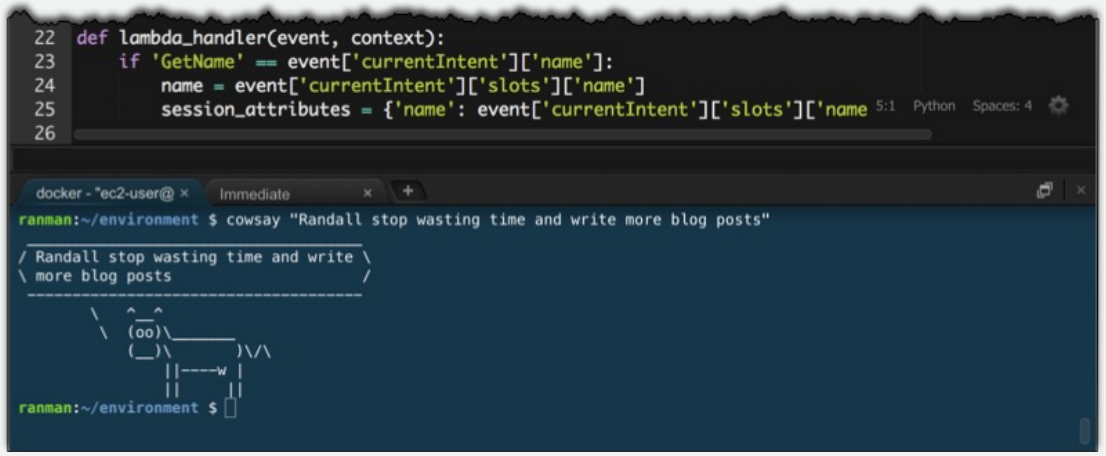
✓ Quickly share your development environment with your team, enabling you to pair program and track each other's inputs in real time.



```python
import boto3
option_table = boto3.resource('dynamodb').Table('options')
vote_table = boto3.resource('dynamodb').Table('votes')
#more cowbell
def build_response(message, message_type="Close", session_attributes=None):
    resp = {
        "dialogAction": {
            "type": message_type,
            "message": {
                "contentType": "PlainText",
                "content": message
            }
        }
    }
    if message_type is 'Close':
        resp["dialogAction"]["fulfillmentState"] = "Fulfilled"
    if session_attributes:
        resp['sessionAttributes'] = session_attributes
    return resp

def lambda_handler(event, context):
    if 'GetName' == event['currentIntent']['name']:
        name = event['currentIntent']['slots']['name']
        session_attributes = {'name': event['currentIntent']['slots']['name']}
        return build_response(
            "Thanks {} you can ask me to describe the episodes".format(name),
            message_type="ElicitIntent",
            session_attributes=session_attributes)
    if 'DescribeEpisodesTwo' == event['currentIntent']['name']:
        options = option_table.get_item(Key={'poll': 'episodes'})['Item']['options']
        msg = ""
        for i, option in enumerate(options):
            msg += "{} {}\n".format(i, option)
        return build_response(msg, message_type="ElicitIntent", session_attributes=event['sessionAttributes'])
    elif 'VoteEpisodeTwo' == event['currentIntent']['name']:
        item = {
            'user': event['userId'],
            'vote': event['currentIntent']['slots']['option'],
            'poll': 'episodes'
        }
        name = event['userId']
        if event['sessionAttributes'].get('name'):
            item['name'] = event['sessionAttributes']['name']
            name = item['name']
        vote_table.put_item(Item=item)
        return build_response(
            "{} voted for {}".format(
                name, event['currentIntent']['slots']['option']),
                session_attributes=event['sessionAttributes']
```

```
ranman:~/environment $ ls
deleteme  episode_voter  README.md
ranman:~/environment $
ranman:~/environment $ ls
deleteme  episode_voter  README.md
ranman:~/environment $
ranman:~/environment $
```



```python
def lambda_handler(event, context):
    if 'GetName' == event['currentIntent']['name']:
        name = event['currentIntent']['slots']['name']
        session_attributes = {'name': event['currentIntent']['slots']['name'
```

```
ranman:~/environment $ cowsay "Randall stop wasting time and write more blog posts"
 __________________________________
/ Randall stop wasting time and write \
\ more blog posts                     /
 -----------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
ranman:~/environment $
```

AWS blog - Cloud9 : https://aws.amazon.com/ko/blogs/korea/aws-cloud9-cloud-developer-environments/

5. IRSA 실습 – 1 (환경 구성)

1. Region 변수 설정

```
$ AWS_REGION = us-west-2
```

2. aws cli 확인 및 account_id 변수 설정

```
$ aws sts get-caller-identity
```

3. 클러스터 OIDC 확인 및 oidc_provider 변수 설정

```
$ aws eks describe-cluster --name eks-workshop --region $AWS_REGION --query "cluster.identity.oidc.issuer" --output text | sed -e
"s/^https:\/\///"
```

4. Namespace 생성

```
$ kubectl apply –f ns.yaml
```

5. IRSA 실습 – 2 (권한 관리)

1. IAM Policy 생성

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "sts:*",
            "eks:*"
        ],
        "Resource": "*"
    }]
}
```

2. Trust Relationship 생성

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::<account-id>:oidc-
provider/oidc.eks.$AWS_REGION.amazonaws.com/id/<xxxxxxxxxxxx>"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.$AWS_REGION.amazonaws.com/id/<xxxxxxxxxxxx>:aud":
"sts.amazonaws.com",
                    "oidc.eks.$AWS_REGION.amazonaws.com/id/<xxxxxxxxxxxx>:sub":
"system:serviceaccount:python:python-test-sa"
                }
            }
        }
    ]
}
```

5. IRSA 실습 – 2 (권한 관리)

3. Role 생성 및 Policy 연결

```
$ aws iam create-role \
  --role-name PythonTestRole \
  --assume-role-policy-document file://trust-relationship.json
```

```
$ aws iam attach-role-policy \
 --policy-arn arn:aws:iam::$account_id:policy/PythonTestRole-policy \
 --role-name PythonTestRole
```

4. ServiceAccount 생성

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: python-test-sa
  namespace: python
  labels:
    app.kubernetes.io/name: python-test
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::$account_id:role/PythonTestRole
automountServiceAccountToken: true
```

```
$ kubectl apply -f sa.yaml
```

```
$ kubectl get sa python-test-sa -n python
NAME            SECRETS   AGE
python-test-sa   0         5m
```

5. IRSA 실습 – 3 (테스트 이미지 생성)

```
from flask import Flask, request, jsonify
import boto3

app = Flask(__name__)


@app.route('/sts', methods=['GET'])
def sts():
   stsClient = boto3.client('sts')
   response = stsClient.get_caller_identity()
   print(response)
   return jsonify(response)


@app.route('/eks', methods=['GET'])
def eksClsuter():
   eksClient = boto3.client('eks')
   response = eksClient.list_clusters()
   print(response)
   return jsonify(response)

if __name__ == '__main__':
app.run(host='0.0.0.0', port=6000, debug=True)
```

```
$ aws ecr create-repository --repository-name python
```

```
$ ls –al
Dockerfile      cicd.sh        main.py        requirements.txt
```

```
$ ./cicd.sh
```

```
$ docker image ls
REPOSITORY                                                TAG       IMAGE ID
CREATED        SIZE
<account-id>.dkr.ecr.<region>.amazonaws.com/python             latest   …
```

```
$ aws ecr list-images --repository-name python
{
   "imageIds": [
      {
         "imageDigest":
"sha256:84b698343137d9829d7a9c7b4549f8cbc8294548ad07f8ea35d6bd01
7db4e324",
         "imageTag": "latest"
      }
   ]
}
```

5. IRSA 실습 – 4 (Pod 생성하기)

```
$ cat deployment-default.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-test-default
  namespace: python
spec:
  selector:
    matchLabels:
      app: python-test
  replicas: 1
  template:
    metadata:
      labels:
        app: python-test
    spec:
      containers:
      - name: python-test
        image:
<account_id>.dkr.ecr.$AWS_REGION.amazonaws.com/python:latest
        ports:
        - containerPort: 6000
```

```
$ cat deployment-irsa.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-test-irsa
  namespace: python
spec:
  selector:
    matchLabels:
      app: python-test
  replicas: 1
  template:
    metadata:
      labels:
        app: python-test
    spec:
      containers:
      serviceAccountName: python-test-sa
      - name: python-test
        image:
<account_id>.dkr.ecr.$AWS_REGION.amazonaws.com/python:latest
        ports:
        - containerPort: 6000
```

5. IRSA 실습 – 4 (Pod 생성하기)

```
$ kubectl describe pod python-test-default-xxx –n python
Name:              python-test-default-xxx
Service Account: default
Containers:
  python-test :
    Image: python:latest
    Port:       6000/TCP
    Environment: <none>
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-np77j (ro)
Volumes:
  kube-api-access-np77j:
    Type:                  Projected (a volume that contains injected data from multiple
sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:          kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:            true
```

```
$ kubectl describe pod python-test-default-xxx –n python
Name:              python-test-default-xxx
Service Account: python-test-sa
Containers:
  python-test :
    Image: python:latest
    Port:       6000/TCP
    Environment:
      AWS_REGION: us-west-2
      AWS_STS_REGIONAL_ENDPOINTS: regional
      AWS_ROLE_ARN: arn:aws:iam::<account-id>:role/<EKS_CUSTOM_ROLE>
      AWS_WEB_IDENTITY_TOKEN_FILE:
/var/run/secrets/eks.amazonaws.com/serviceaccount/token
    Mounts:
      /var/run/secrets/eks.amazonaws.com/serviceaccount from aws-iam-token (ro)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-np77j (ro)
Volumes:
  aws-iam-token:
    Type:                  Projected (a volume that contains injected data from multiple
sources)
    TokenExpirationSeconds:  86400
  kube-api-access-np77j:
    Type:                  Projected (a volume that contains injected data from multiple
sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:          kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:            true
```

5. IRSA 실습 – 4 (Pod 생성하기)

```
$ kubectl debug python-test-default-xxx -n python -it --
image=nicolaka/netshoot

python-test-default-xxx> $ curl 127.0.0.1:6000/sts
{
  "Arn": "arn:aws:sts::<account-id>:assumed-
role/<NodeInstanceRole>/<node-instance>",
  ...
}

python-test-default-xxx> $ curl 127.0.0.1:6000/eks
```

```
$ kubectl debug python-test-irsa-xxx -n python -it --
image=nicolaka/netshoot

python-test-irsa-xxx> $ curl 127.0.0.1:6000/sts
{
  "Arn": "arn:aws:sts::<account-id>:assumed-role/PythonTestRole/botocore-
session-xxx",
  ...
}

python-test-irsa-xxx> $ curl 127.0.0.1:6000/eks
```

5. IRSA 실습 - CloudTrail

CloudTrail

- AssumeRoleWithWebIdentity
- GetCallerIdentity
- ListClusters

```
{
   "eventVersion": "1.08",
   "userIdentity": {
      "type": "WebIdentityUser",
      "userName": "system:serviceaccount:python:python-test-sa",
      "identityProvider": "arn:aws:iam::<account-id>:oidc-provider/oidc.eks.<region>.amazonaws.com/id/45B3BC539A5A37E3A4091367C02D8FB1"
   },
   "eventName": "AssumeRoleWithWebIdentity",
   "requestParameters": {
      "roleArn": "arn:aws:iam::<account-id>:role/PythonTestRole",
      "roleSessionName": "botocore-session-1700378255"
   },
   "responseElements": {
      "credentials": {
         "accessKeyId": "ASIAXAOPZ562R5TEO2U3",
         "sessionToken": ”<session-token>”
      },
      "subjectFromWebIdentityToken": "system:serviceaccount:python:python-test-sa",
      "assumedRoleUser": {
         "assumedRoleId": "AROAXAOPZ562ZL5RXPPKQ:botocore-session-1700378255",
         "arn": "arn:aws:sts::<account-id>:assumed-role/PythonTestRole/botocore-session-1700378255"
      },
      "provider": "arn:aws:iam::<account-id>:oidc-provider/oidc.eks.<region>.amazonaws.com/id/45B3BC539A5A37E3A4091367C02D8FB1",
      "audience": "sts.amazonaws.com"
   },…
```

5. IRSA 실습 - CloudTrail

CloudTrail

- AssumeRoleWithWebIdentity
- GetCallerIdentity
- ListClusters

```
{
   "eventVersion": "1.08",
   "userIdentity": {
      "type": "AssumedRole",
      "principalId": "AROAXAOPZ562ZL5RXPPKQ:botocore-session-1700378255",
      "arn": "arn:aws:sts::<account-id>:assumed-role/PythonTestRole/botocore-session-1700378255",
      "accountId": "<account-id>",
      "accessKeyId": "ASIAXAOPZ562R5TEO2U3",
      "sessionContext": {
         "sessionIssuer": {
            "type": "Role",
            "principalId": "AROAXAOPZ562ZL5RXPPKQ",
            "arn": "arn:aws:iam::<account-id>:role/PythonTestRole",
            "accountId": "<account-id>",
            "userName": "PythonTestRole"
         },
         "webIdFederationData": {
            "federatedProvider": "arn:aws:iam::<account-id>:oidc-provider/oidc.eks.<region>.amazonaws.com/id/45B3BC539A5A37E3A4091367C02D8FB1",
            "attributes": {}
         }
      }
   },
…
```

Q&A

Thank you!

고병수

이수정